STAT 139: Final Project

Danny Kim, Christopher Lee, Karina Wang, Daniel Son

2022-12-14

EDA

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Load team data
team_data = list()
team_wins <- list()
drop = c("W", "L")
for (year in 1997:2022) {
  df1 = read.csv(paste("data/teams_data/batting", year, ".csv", sep=""))
  df2 = read.csv(paste("data/teams_data/pitching", year, ".csv", sep=""))
  df3 = read.csv(paste("data/teams_data/fielding", year, ".csv", sep=""))
  df_tot = merge(merge(df1, df2, by="Tm", suffixes=c(".bat", ".pitch")), df3, by="Tm", suffixes=c("", "

  df_tot = df_tot[
    !(df_tot$Tm %in% c("", "League Average")),
    !(names(df_tot) %in% drop)
  ]
  df_tot$Tm = factor(df_tot$Tm)
  team_data[[year]] = df_tot
  team_wins[[year]] = df_tot[, c("Tm", "W.L.")]
}

# Load player data
years <- 1997:2022
bps <- c("batting", "pitching", "fielding")
player_data <- list()
for (year in years) {
```

```
  player_data[[year]] <- list()
  for (bp in bps) {
    player_data[[year]][[bp]] <- read.csv(paste("data/player_data/", bp, year, ".csv", sep=""))
    quant_cols <- names(select_if(player_data[[year]][[bp]], is.numeric))
    for (col in quant_cols) {
      # impute data with mean
      df <- player_data[[year]][[bp]]
      player_data[[year]][[bp]][is.na(player_data[[year]][[bp]][,col]),col] <- mean(df[,col], na.rm=TRU
    }
  }
}

fa_data = list()
for (year in years) {
  fa_data[[year]] = read.csv(paste("data/fa_data/fa", year, ".csv", sep=""))
  fa_data[[year]]$WAR3[is.na(fa_data[[year]]$WAR3)] = 0
}
```

```
# Data Cleaning for the Team Data
team_wins <- list()
for (year in years) {
  team_wins[[year]] <- team_data[[year]][!(team_data[[year]]$Tm %in% c("", "League Average")), c("Tm", 
}
```

```
# Clean player data
for (year in years) {
  for (bp in bps) {
    player_data[[year]][[bp]]$year <- year
    player_data[[year]][[bp]]$year_adj <- year - 1997
  }
}

for (year in years) {
  player_data[[year]][["pitching"]] = player_data[[year]][["pitching"]][!is.infinite(player_data[[year]]
}

long_team_names <- team_data[[year]][!(team_data[[year]]$Tm %in% c("", "League Average")),]$Tm
short_team_names <- c("ARI", "ATL", "BAL", "BOS", "CHC", "CHW", "CIN", "CLE", "COL", "DET",
                      "HOU", "KCR", "LAA", "LAD", "MIA", "MIL", "MIN", "NYM", "NYY", "OAK",
                      "PHI", "PIT", "SDP", "SFG", "SEA", "STL", "TBR", "TEX", "TOR", "WSN")
agg_data <- list()
for (year in years) {
  agg_data[[year]] <- list()
  for (bp in bps) {
    quant_cols <- names(select_if(player_data[[year]][[bp]], is.numeric))
    agg_data[[year]][[bp]] <- player_data[[year]][[bp]][, c("Tm", quant_cols)] %>%
      group_by(Tm) %>%
      summarise(across(quant_cols, ~weighted.mean(., w = G)))
    agg_data[[year]][[bp]] <- agg_data[[year]][[bp]][!(agg_data[[year]][[bp]]$Tm == "TOT"),]
    agg_data[[year]][[bp]]$long_Tm <- factor(
      agg_data[[year]][[bp]]$Tm,
      levels=short_team_names,
      labels=long_team_names
```

```
    )
  }
}
```

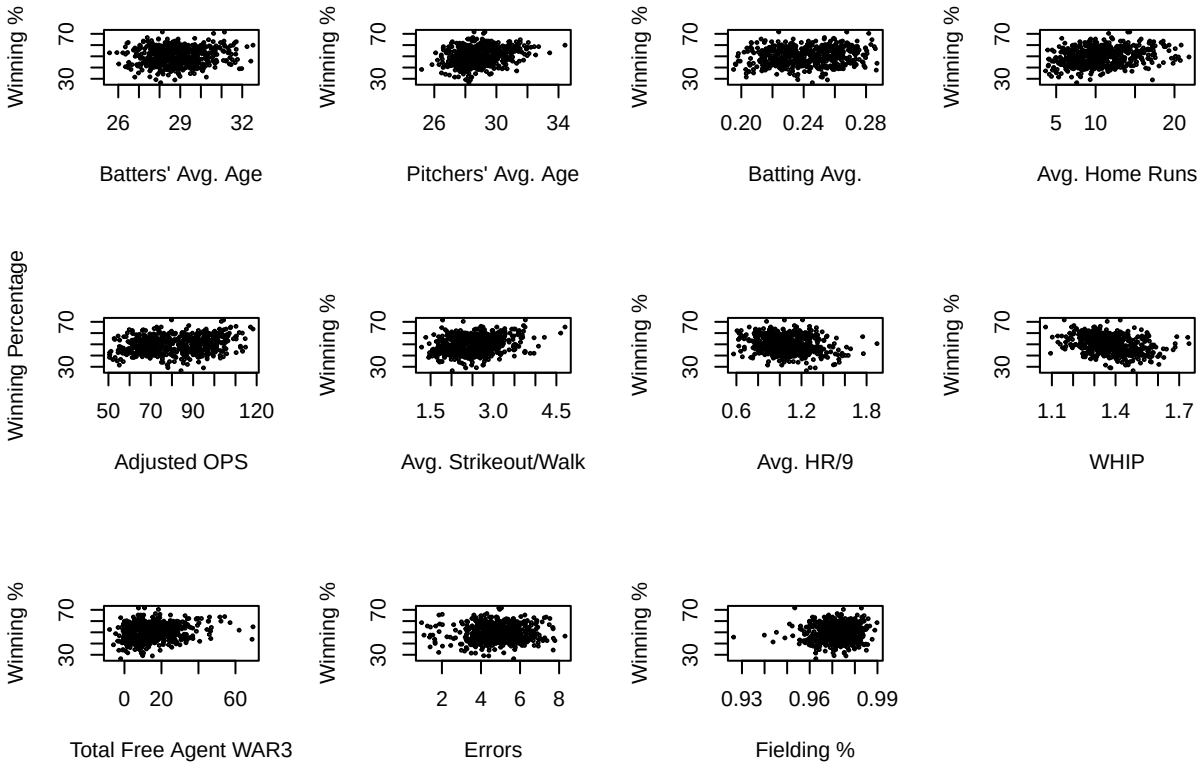
```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use ‘all_of()‘ or ‘any_of()‘ instead.
##   # Was:
##   data %>% select(quant_cols)
##
##   # Now:
##   data %>% select(all_of(quant_cols))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

```
player_combo <- list()
for (year in years) {
  player_combo[[year]] <- merge(merge(agg_data[[year]][[bps[1]]], agg_data[[year]][[bps[2]]], by="Tm", s
}

agg_fa_data <- list()
for (year in years) {
  agg_fa_data[[year]] = fa_data[[year]] %>% group_by(To.Team) %>% summarise(tot_fa_war3=sum(WAR3), num_
}

# add response variable to player data
player_with_wins <- list()
for (year in 1997:2021) {
  player_with_wins[[year]] <- merge(player_combo[[year]], team_wins[[year+1]], by.x="long_Tm.pitch", by
}

player_with_wins_fa <- list()
for (year in 1997:2021) {
  player_with_wins_fa[[year]] <- merge(player_with_wins[[year]], agg_fa_data[[year]], by.x="long_Tm.pit
}


player_with_wins_combined = bind_rows(player_with_wins_fa, )
player_with_wins_combined$W.L..same_year = 100 * player_with_wins_combined$W.L..same_year
player_with_wins_combined$W.L..next_year = 100 * player_with_wins_combined$W.L..next_year

drop_cols = c("long_Tm.pitch", "Rk.bat", "G.bat", "long_Tm.bat", "Rk.pitch", "W", "L", "G.pitch", "long_
              "Age", "GS", "CG", "GS.field", "CG.field", "Rdrs", "Rdrs.yr", "Rgood")
player_with_wins_combined = player_with_wins_combined[, !(names(player_with_wins_combined) %in% drop_col

n.rows = nrow(player_with_wins_combined)
n.train = 0.8 * n.rows
train.rows = sample(n.rows, n.train)
train.df = player_with_wins_combined[train.rows,]
colnames(train.df)[colnames(train.df) == 'OPS.'] <- 'OPSplus'
colnames(train.df)[colnames(train.df) == 'ERA.'] <- 'ERAplus'
test.df = player_with_wins_combined[-train.rows,]
colnames(test.df)[colnames(test.df) == 'OPS.'] <- 'OPSplus'
colnames(test.df)[colnames(test.df) == 'ERA.'] <- 'ERAplus'
```

```r
# train.df
names(train.df)
```

```
##  [1] "Tm"              "Age.bat"         "PA"              "AB"
##  [5] "R.bat"           "H.bat"           "X2B"             "X3B"
##  [9] "HR.bat"          "RBI"             "SB"              "CS"
## [13] "BB.bat"          "SO.bat"          "BA"              "OBP"
## [17] "SLG"             "OPS"             "OPSplus"         "TB"
## [21] "GDP"             "HBP.bat"         "SH"              "SF"
## [25] "IBB.bat"         "year.bat"        "year_adj.bat"    "Age.pitch"
## [29] "W.L..same_year" "ERA"             "GF"              "SHO"
## [33] "SV"              "IP"              "H.pitch"         "R.pitch"
## [37] "ER"              "HR.pitch"        "BB.pitch"        "IBB.pitch"
## [41] "SO.pitch"        "HBP.pitch"       "BK"              "WP"
## [45] "BF"              "ERAplus"         "FIP"             "WHIP"
## [49] "H9"              "HR9"             "BB9"             "SO9"
## [53] "SO.W"            "year.pitch"      "year_adj.pitch" "Rk"
## [57] "G"               "Inn"             "Ch"              "PO"
## [61] "A"               "E"               "DP"              "Fld."
## [65] "Rtot"            "Rtot.yr"         "RF.9"            "RF.G"
## [69] "year"            "year_adj"        "W.L..next_year" "tot_fa_war3"
## [73] "num_fas"
```

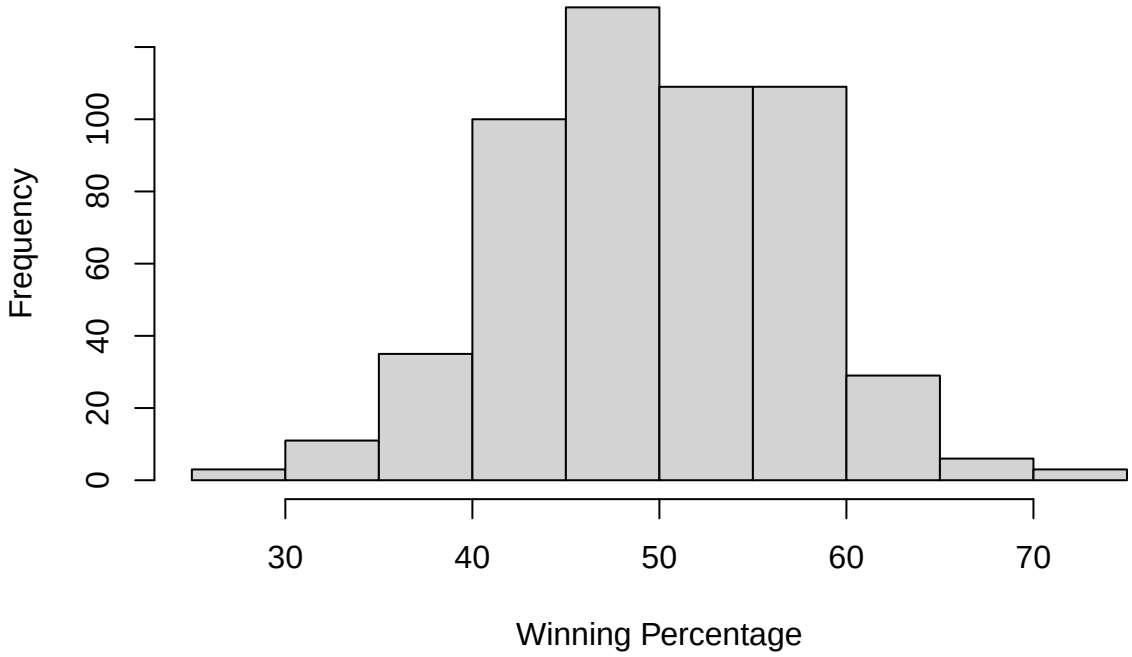
```r
# Explore Potential Predictors
par(mfrow=c(3,4))
plot(W.L..next_year ~ Age.bat, data=train.df,
     xlab="Batters' Avg. Age", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ Age.pitch, data=train.df,
     xlab="Pitchers' Avg. Age", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ BA, data=train.df,
     xlab="Batting Avg.", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ HR.bat, data=train.df,
     xlab="Avg. Home Runs", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ OPSplus, data=train.df,
     xlab="Adjusted OPS", ylab="Winning Percentage", cex=0.3)
plot(W.L..next_year ~ SO.W, data=train.df,
     xlab="Avg. Strikeout/Walk", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ HR9, data=train.df,
     xlab="Avg. HR/9", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ WHIP, data=train.df,
     xlab="WHIP", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ tot_fa_war3, data=train.df,
     xlab="Total Free Agent WAR3", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ E, data=train.df,
     xlab="Errors", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ Fld., data=train.df,
     xlab="Fielding %", ylab="Winning %", cex=0.3)
```




```
# Summary statistics for winpct
summary(train.df$W.L..next_year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   26.50   43.80   49.40   49.84   55.60   71.70
```

```
# Histogram for winpct
hist(train.df$W.L..next_year, main="Distribution of Winning Percentage",
     xlab="Winning Percentage")
```




```
# Correlation matrix
cor(train.df[, c("W.L..next_year", "Age.bat", "Age.pitch", "BA", "HR.bat", "OPS", "SO.W", "HR9", "WHIP"
```

```
##                 W.L..next_year     Age.bat   Age.pitch          BA      HR.bat
## W.L..next_year      1.00000000  0.11060537  0.24337623  0.14154523  0.25102094
## Age.bat             0.11060537  1.00000000  0.55832567  0.18116617  0.13513857
## Age.pitch           0.24337623  0.55832567  1.00000000  0.08582083  0.18445521
## BA                  0.14154523  0.18116617  0.08582083  1.00000000  0.53138458
## HR.bat              0.25102094  0.13513857  0.18445521  0.53138458  1.00000000
## OPS                 0.22529744  0.15300539  0.13823860  0.91695295  0.70605134
## SO.W                0.24826183 -0.10548820  0.04857586 -0.12754165  0.07317230
## HR9                -0.22578978 -0.14827185 -0.11855328  0.05248312  0.10836314
## WHIP               -0.33712687 -0.01744970 -0.12810995  0.14170757 -0.05308393
## tot_fa_war3         0.23861857  0.16936155  0.23050266  0.08887711  0.11818638
## E                   0.06018314  0.01923862  0.07862286  0.14071381  0.24242466
## Fld.                0.08106713  0.18367906  0.19057409  0.04776392  0.12742531
##                          OPS        SO.W         HR9        WHIP tot_fa_war3
## W.L..next_year  0.22529744  0.24826183 -0.22578978 -0.33712687  0.23861857
## Age.bat         0.15300539 -0.10548820 -0.14827185 -0.01744970  0.16936155
## Age.pitch       0.13823860  0.04857586 -0.11855328 -0.12810995  0.23050266
## BA              0.91695295 -0.12754165  0.05248312  0.14170757  0.08887711
## HR.bat          0.70605134  0.07317230  0.10836314 -0.05308393  0.11818638
## OPS             1.00000000 -0.02376657  0.18307624  0.09919776  0.14617038
## SO.W           -0.02376657  1.00000000 -0.07924629 -0.74129011  0.11314833
## HR9             0.18307624 -0.07924629  1.00000000  0.43887621  0.01106407
## WHIP            0.09919776 -0.74129011  0.43887621  1.00000000 -0.07454616
```

```
## tot_fa_war3     0.14617038  0.11314833  0.01106407 -0.07454616  1.00000000
## E               0.06265711 -0.31973713 -0.17957658  0.18075632 -0.07754190
## Fld.            0.02544533  0.05842458 -0.15459663 -0.17259556  0.05769343
##                          E        Fld.
## W.L..next_year  0.06018314  0.08106713
## Age.bat         0.01923862  0.18367906
## Age.pitch       0.07862286  0.19057409
## BA              0.14071381  0.04776392
## HR.bat          0.24242466  0.12742531
## OPS             0.06265711  0.02544533
## SO.W           -0.31973713  0.05842458
## HR9            -0.17957658 -0.15459663
## WHIP            0.18075632 -0.17259556
## tot_fa_war3    -0.07754190  0.05769343
## E               1.00000000 -0.06447889
## Fld.           -0.06447889  1.00000000
```

```
cor(train.df[, c("W.L..next_year", "Age.bat", "Age.pitch", "BA", "HR.bat", "OPS", "SO.W", "HR9", "WHIP"
```

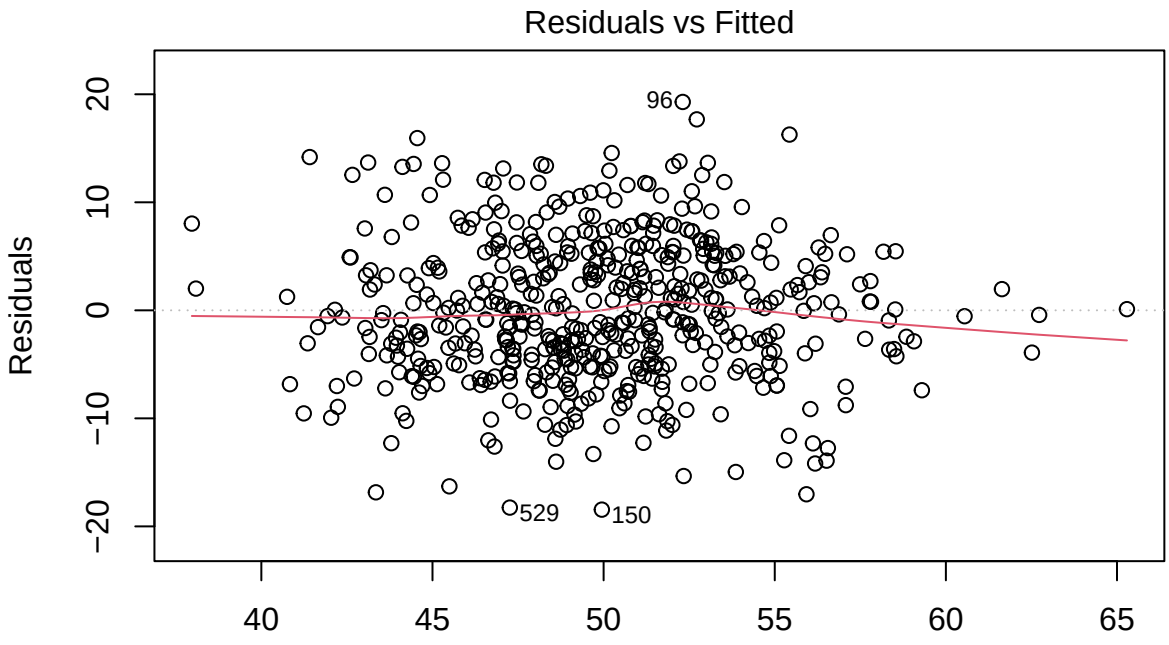
```
##                W.L..next_year      Age.bat   Age.pitch          BA      HR.bat
## W.L..next_year    1.000000000 0.0122335475 0.059231988 0.020035052 0.063011513
## Age.bat           0.012233548 1.0000000000 0.311727557 0.032821182 0.018262434
## Age.pitch         0.059231988 0.3117275565 1.000000000 0.007365215 0.034023724
## BA                0.020035052 0.0328211821 0.007365215 1.000000000 0.282369572
## HR.bat            0.063011513 0.0182624340 0.034023724 0.282369572 1.000000000
## OPS               0.050758934 0.0234106487 0.019109911 0.840802711 0.498508500
## SO.W              0.061633934 0.0111277608 0.002359615 0.016266873 0.005354186
## HR9               0.050981027 0.0219845411 0.014054879 0.002754478 0.011742571
## WHIP              0.113654527 0.0003044919 0.016412159 0.020081035 0.002817904
## tot_fa_war3       0.056938822 0.0286833338 0.053131476 0.007899141 0.013968021
## E                 0.003622010 0.0003701245 0.006181554 0.019800375 0.058769715
## Fld.              0.006571879 0.0337379972 0.036318485 0.002281392 0.016237209
##                         OPS         SO.W          HR9         WHIP  tot_fa_war3
## W.L..next_year 0.0507589344 0.0616339339 0.0509810266 0.1136545270 0.0569388217
## Age.bat        0.0234106487 0.0111277608 0.0219845411 0.0003044919 0.0286833338
## Age.pitch      0.0191099114 0.0023596145 0.0140548792 0.0164121594 0.0531314759
## BA             0.8408027106 0.0162668729 0.0027544784 0.0200810351 0.0078991407
## HR.bat         0.4985084997 0.0053541861 0.0117425707 0.0028179038 0.0139680211
## OPS            1.0000000000 0.0005648498 0.0335169095 0.0098401965 0.0213657787
## SO.W           0.0005648498 1.0000000000 0.0062799752 0.5495110283 0.0128025441
## HR9            0.0335169095 0.0062799752 1.0000000000 0.1926123240 0.0001224137
## WHIP           0.0098401965 0.5495110283 0.1926123240 1.0000000000 0.0055571292
## tot_fa_war3    0.0213657787 0.0128025441 0.0001224137 0.0055571292 1.0000000000
## E              0.0039259134 0.1022318341 0.0322477467 0.0326728487 0.0060127457
## Fld.           0.0006474648 0.0034134318 0.0239001172 0.0297892281 0.0033285317
##                           E         Fld.
## W.L..next_year 0.0036220102 0.0065718789
## Age.bat        0.0003701245 0.0337379972
## Age.pitch      0.0061815539 0.0363184852
## BA             0.0198003754 0.0022813925
## HR.bat         0.0587697146 0.0162372093
## OPS            0.0039259134 0.0006474648
## SO.W           0.1022318341 0.0034134318
## HR9            0.0322477467 0.0239001172
```

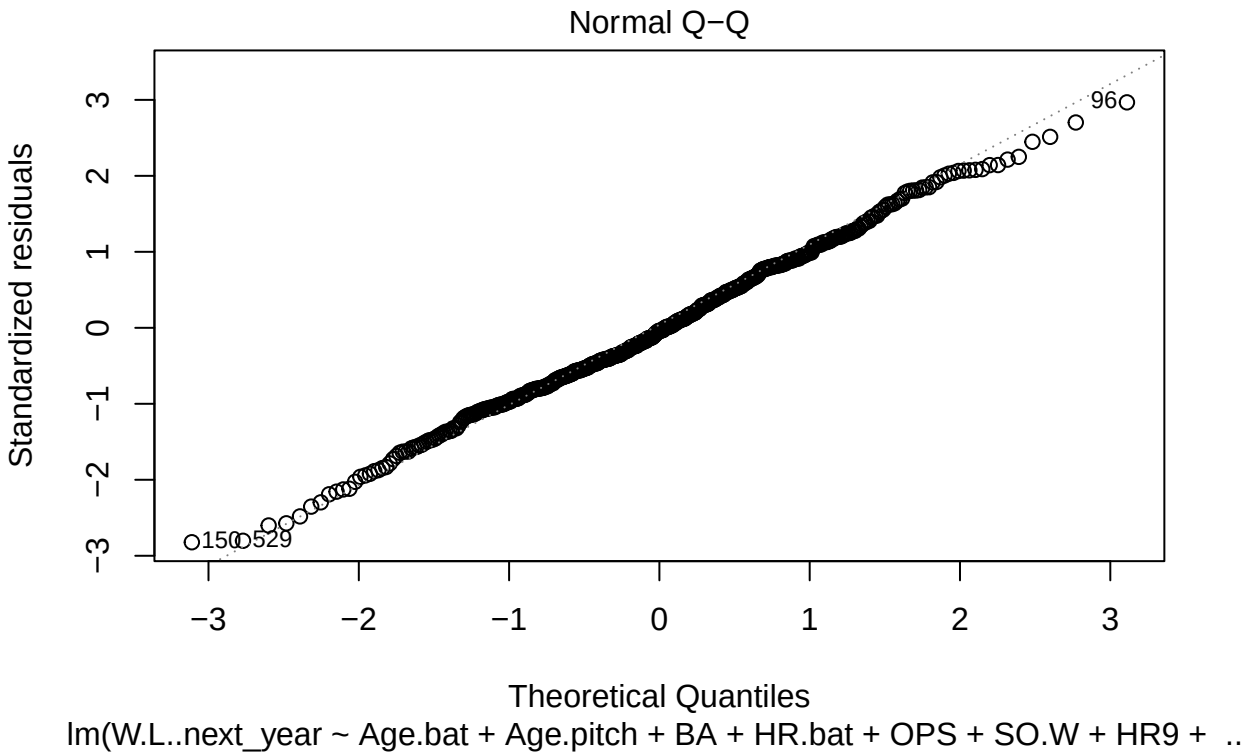
```
## WHIP           0.0326728487 0.0297892281
## tot_fa_war3    0.0060127457 0.0033285317
## E              1.0000000000 0.0041575277
## Fld.           0.0041575277 1.0000000000
```

```
# Baseline Multiple Regression Model
baseline <- lm(W.L..next_year ~ Age.bat + Age.pitch + BA + HR.bat +
                OPS + SO.W + HR9 + WHIP + tot_fa_war3 + E + Fld. , data = train.df)
summary(baseline)
```

```
##
## Call:
## lm(formula = W.L..next_year ~ Age.bat + Age.pitch + BA + HR.bat +
##     OPS + SO.W + HR9 + WHIP + tot_fa_war3 + E + Fld., data = train.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.4487  -4.3667  -0.2621   4.8997  19.2881
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   33.96531   40.30613   0.843 0.399790
## Age.bat       -0.14731    0.29948  -0.492 0.623017
## Age.pitch      0.65735    0.28543   2.303 0.021670 *
## BA          -187.93521   45.95939  -4.089 5.01e-05 ***
## HR.bat        -0.16018    0.14402  -1.112 0.266557
## OPS           96.04057   18.77113   5.116 4.38e-07 ***
## SO.W           0.54093    0.88207   0.613 0.539975
## HR9           -7.30899    1.85640  -3.937 9.36e-05 ***
## WHIP         -16.44353    4.96630  -3.311 0.000994 ***
## tot_fa_war3    0.09777    0.02526   3.870 0.000123 ***
## E              0.78625    0.29394   2.675 0.007710 **
## Fld.           6.04365   39.47920   0.153 0.878391
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.593 on 524 degrees of freedom
## Multiple R-squared:  0.2783, Adjusted R-squared:  0.2631
## F-statistic: 18.37 on 11 and 524 DF,  p-value: < 2.2e-16
```

```
# Asssess Linear Model Assumptions
plot(baseline, which=c(1,2))
```


Residuals vs Fitted
Residuals
Fitted values
lm(W.L..next_year ~ Age.bat + Age.pitch + BA + HR.bat + OPS + SO.W + HR9 + ..
96
529
150
20
10
0
−10
−20
40
45
50
55
60
65




```
RMSE <- function(y,yhat){
  SSE = sum((y-yhat)^2)
  SST = sum((y - mean(y))^2)
  return(sqrt(SSE/length(y)))
}

R2 <- function(y,yhat) {
  SSE = sum((y-yhat)^2)
  SST = sum((y-mean(y))^2)
  r.squared <- 1 - (SSE / SST)
  return(r.squared)
}
baseline.trainRMSE = RMSE(train.df$W.L..next_year, predict(baseline, newdata=train.df))
baseline.testRMSE = RMSE(test.df$W.L..next_year, predict(baseline, newdata=test.df))
baseline.trainR2 = R2(train.df$W.L..next_year, predict(baseline, newdata=train.df))
baseline.testR2 = R2(test.df$W.L..next_year, predict(baseline, newdata=test.df))
```

Linear Regression

```
colnames(train.df)
```

```
##  [1] "Tm"             "Age.bat"        "PA"             "AB"
##  [5] "R.bat"          "H.bat"          "X2B"            "X3B"
```

```
##  [9] "HR.bat"         "RBI"            "SB"             "CS"
## [13] "BB.bat"         "SO.bat"         "BA"             "OBP"
## [17] "SLG"            "OPS"            "OPSplus"        "TB"
## [21] "GDP"            "HBP.bat"        "SH"             "SF"
## [25] "IBB.bat"        "year.bat"       "year_adj.bat"   "Age.pitch"
## [29] "W.L..same_year" "ERA"            "GF"             "SHO"
## [33] "SV"             "IP"             "H.pitch"        "R.pitch"
## [37] "ER"             "HR.pitch"       "BB.pitch"       "IBB.pitch"
## [41] "SO.pitch"       "HBP.pitch"      "BK"             "WP"
## [45] "BF"             "ERAplus"        "FIP"            "WHIP"
## [49] "H9"             "HR9"            "BB9"            "SO9"
## [53] "SO.W"           "year.pitch"     "year_adj.pitch" "Rk"
## [57] "G"              "Inn"            "Ch"             "PO"
## [61] "A"              "E"              "DP"             "Fld."
## [65] "Rtot"           "Rtot.yr"        "RF.9"           "RF.G"
## [69] "year"           "year_adj"       "W.L..next_year" "tot_fa_war3"
## [73] "num_fas"
```

```
# full linear regression models

# ignore Rk.bat, R.bat, RBI, year.bat, year_adj.bat, W, L, R.pitch, year.pitch
# year_adj.pitch, Rk, WL..next_year, year, year_adj, ERA, ERAplus
# Rtot, Rtot.yr, Rdrs, Rgood (hard to interpret)
lm.full <- lm(W.L..next_year ~ Age.bat + PA + AB + H.bat + X2B + X3B +
                HR.bat + SB + CS + BB.bat + SO.bat + BA + OBP + SLG + OPS + OPSplus +
                TB + GDP + HBP.bat + SH + SF + IBB.bat + Age.pitch + W.L..same_year +
                GF + SHO + SV + IP + H.pitch + HR.pitch +
                BB.pitch + IBB.pitch + SO.pitch + HBP.pitch + BK + WP + BF +
                FIP + WHIP + H9 + HR9 + BB9 + SO9 + SO.W +
                G + Inn + Ch + PO + A + E + DP + Fld. +
                RF.9 + RF.G + tot_fa_war3 + num_fas,
              data = train.df)
lmfull.trainRMSE = RMSE(train.df$W.L..next_year, predict(lm.full, newdata=train.df))
```

```
## Warning in predict.lm(lm.full, newdata = train.df): prediction from a rank-
## deficient fit may be misleading
```

```
lmfull.testRMSE = RMSE(test.df$W.L..next_year, predict(lm.full, newdata=test.df))
```

```
## Warning in predict.lm(lm.full, newdata = test.df): prediction from a rank-
## deficient fit may be misleading
```

```
lmfull.trainR2 = R2(train.df$W.L..next_year, predict(lm.full, newdata=train.df))
```

```
## Warning in predict.lm(lm.full, newdata = train.df): prediction from a rank-
## deficient fit may be misleading
```

```
lmfull.testR2 = R2(test.df$W.L..next_year, predict(lm.full, newdata=test.df))
```

```
## Warning in predict.lm(lm.full, newdata = test.df): prediction from a rank-
## deficient fit may be misleading
```

```
lm.fullinteraction <- lm(W.L..next_year ~ (Age.bat + PA + AB + H.bat + X2B + X3B +
                HR.bat + SB + CS + BB.bat + SO.bat + BA + OBP + SLG + OPS + OPSplus +
                TB + GDP + HBP.bat + SH + SF + IBB.bat + Age.pitch + W.L..same_year +
                GF + SHO + SV + IP + H.pitch + HR.pitch +
                BB.pitch + IBB.pitch + SO.pitch + HBP.pitch + BK + WP + BF +
                FIP + WHIP + H9 + HR9 + BB9 + SO9 + SO.W +
                G + Inn + Ch + PO + A + E + DP + Fld. +
                RF.9 + RF.G + tot_fa_war3 + num_fas)^2, data = train.df)
lmfullinteraction.trainRMSE = RMSE(train.df$W.L..next_year, predict(lm.fullinteraction, newdata=train.d
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = train.df): prediction from a
## rank-deficient fit may be misleading
```

```
lmfullinteraction.testRMSE = RMSE(test.df$W.L..next_year, predict(lm.fullinteraction, newdata=test.df))
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = test.df): prediction from a
## rank-deficient fit may be misleading
```

```
lmfullinteraction.trainR2 = R2(train.df$W.L..next_year, predict(lm.fullinteraction, newdata=train.df))
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = train.df): prediction from a
## rank-deficient fit may be misleading
```

```
lmfullinteraction.testR2 = R2(test.df$W.L..next_year, predict(lm.fullinteraction, newdata=test.df))
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = test.df): prediction from a
## rank-deficient fit may be misleading
```

```
# Ridge Regression
set.seed(139)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.2.2
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-4
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked _by_ '.GlobalEnv':
##
##     R2, RMSE
```

```r
# regularize full model
X.full = model.matrix(lm.full)[,-1] # drop intercept
best_lambda = cv.glmnet(X.full, train.df$W.L..next_year, alpha=0, lambda=10^seq(-4, 4, 0.1))$lambda.min
ridges.full = glmnet(X.full, train.df$W.L..next_year, alpha=0,
                     lambda=best_lambda)
varImp(ridges.full, lambda=best_lambda)
```

```
##                     Overall
## Age.bat        5.289319e-01
## PA             4.216334e-03
## AB             6.931998e-03
## H.bat          3.409880e-03
## X2B            2.263317e-02
## X3B            9.692054e-01
## HR.bat         1.887635e-01
## SB             1.462568e-02
## CS             3.104388e-02
## BB.bat         1.253634e-01
## SO.bat         3.596029e-02
## BA             1.363215e+00
## OBP            1.915635e+01
## SLG            1.518916e+01
## OPS            1.012715e+01
## OPSplus        8.402160e-03
## TB             4.244888e-03
## GDP            2.731856e-01
## HBP.bat        2.929644e-02
## SH             1.421032e-01
## SF             1.165434e-01
## IBB.bat        6.822091e-01
## Age.pitch      4.097178e-01
## W.L..same_year 5.023869e-02
## GF             1.621340e-01
## SHO            5.412661e+00
## SV             2.196924e-01
## IP             4.886602e-03
## H.pitch        1.956010e-02
## HR.pitch       2.171649e-01
## BB.pitch       7.580050e-03
## IBB.pitch      2.470673e-01
## SO.pitch       3.112174e-02
## HBP.pitch      6.395463e-03
## BK             1.195930e+00
## WP             2.416371e-01
## BF             3.970605e-04
## FIP            7.461530e-01
## WHIP           4.464753e+00
## H9             1.002166e+00
## HR9            2.137497e+00
## BB9            4.326087e-02
## SO9            2.538087e-01
## SO.W           6.312821e-01
## G              2.048192e-02
```

```
## Inn             3.475591e-03
## Ch              1.686871e-03
## PO              1.098725e-03
## A               8.625063e-03
## E               4.172024e-01
## DP              4.778159e-02
## Fld.            4.533977e+00
## RF.9            1.036897e+00
## RF.G            6.774636e-01
## tot_fa_war3     9.617085e-02
## num_fas         1.390332e-01
```

```
X.full.test = model.matrix(lm.full, data=test.df)[,-1] # drop intercept

yhats.full.train = predict(ridges.full, X.full)
ridgesfull.trainRMSE = RMSE(train.df$W.L..next_year, yhats.full.train) # train RMSE
ridgesfull.trainR2 = R2(train.df$W.L..next_year, yhats.full.train) # train R2

yhats.full.test = predict(ridges.full, X.full.test)
#plot(RMSE.ridges.full.test~log(ridges.full$lambda, 10), type='l')
ridgesfull.testRMSE = RMSE(test.df$W.L..next_year, yhats.full.test) # test RMSE
ridgesfull.testR2 = R2(test.df$W.L..next_year, yhats.full.test) # test R2
```

```
set.seed(139)
# regularize full interaction model
X.fullinteraction = model.matrix(lm.fullinteraction)[,-1] # drop intercept

best_lambda = cv.glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=0,
                        lambda=10^seq(-4, 4, 0.1))$lambda.min
ridges.fullinteraction = glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=0,
                      lambda=best_lambda)
X.fullinteraction.test = model.matrix(lm.fullinteraction, data=test.df)[,-1] # drop intercept

yhats.fullinteraction.train = predict(ridges.fullinteraction, X.fullinteraction)
ridgesfullinteraction.trainRMSE = RMSE(train.df$W.L..next_year, yhats.fullinteraction.train) # train RM
ridgesfullinteraction.trainR2 = R2(train.df$W.L..next_year, yhats.fullinteraction.train) # train R2

yhats.fullinteraction.test = predict(ridges.fullinteraction, X.fullinteraction.test)
#plot(RMSE.ridges.fullinteraction.test~log(ridges.fullinteraction$lambda, 10), type='l')
ridgesfullinteraction.testRMSE = RMSE(test.df$W.L..next_year, yhats.fullinteraction.test) # train RMSE
ridgesfullinteraction.testR2 = R2(test.df$W.L..next_year, yhats.fullinteraction.test) # train R2
```

```
# Lasso Regression
# regularize full model
set.seed(139)
best_lambda = cv.glmnet(X.full, train.df$W.L..next_year, alpha=1,
                        lambda=10^seq(-4, 4, 0.1))$lambda.min
lassos.full = glmnet(X.full, train.df$W.L..next_year, alpha=1,
                      lambda=best_lambda)
varImp(lassos.full, lambda=best_lambda)
```

```
##                   Overall
## Age.bat         0.66725103
```

```
## PA              0.00000000
## AB              0.01245382
## H.bat           0.00000000
## X2B             0.00000000
## X3B             1.07397179
## HR.bat          0.19941780
## SB              0.00000000
## CS              0.00000000
## BB.bat          0.20517362
## SO.bat          0.06060037
## BA              0.00000000
## OBP             0.00000000
## SLG            19.44122139
## OPS            11.47807956
## OPSplus         0.00000000
## TB              0.00000000
## GDP             0.29610040
## HBP.bat         0.00000000
## SH              0.00000000
## SF              0.00000000
## IBB.bat         0.70959428
## Age.pitch       0.46895177
## W.L..same_year  0.03107421
## GF              0.08243106
## SHO             4.86147015
## SV              0.12408025
## IP              0.00000000
## H.pitch         0.00000000
## HR.pitch        0.30167177
## BB.pitch        0.00000000
## IBB.pitch       0.08749174
## SO.pitch        0.02095959
## HBP.pitch       0.00000000
## BK              0.45209135
## WP              0.09418293
## BF              0.00000000
## FIP             1.63412645
## WHIP            0.00000000
## H9              1.94643066
## HR9             0.45282890
## BB9             0.00000000
## SO9             0.00000000
## SO.W            0.73015619
## G               0.00000000
## Inn             0.00461364
## Ch              0.00000000
## PO              0.00000000
## A               0.00000000
## E               0.45395431
## DP              0.00000000
## Fld.            0.00000000
## RF.9            0.00000000
## RF.G            0.00000000
## tot_fa_war3     0.11369928
```

```
## num_fas         0.15883053
```

```
yhats.full.train = predict(lassos.full, X.full)
lassosfull.trainRMSE = RMSE(train.df$W.L..next_year, yhats.full.train) # train RMSE
lassosfull.trainR2 = R2(train.df$W.L..next_year, yhats.full.train) # train R2

yhats.full.test = predict(lassos.full, X.full.test)
#plot(RMSE.lassos.full.test~log(ridges.full$lambda, 10), type='l')
lassosfull.testRMSE = RMSE(test.df$W.L..next_year, yhats.full.test) # test RMSE
lassosfull.testR2 = R2(test.df$W.L..next_year, yhats.full.test) # test RMSE
```

```
# regularize full interaction model
set.seed(139)
best_lambda = cv.glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=1,
                        lambda=10^seq(-4, 4, 0.1))$lambda.min
lassos.fullinteraction = glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=1,
                      lambda=best_lambda)
yhats.fullinteraction.train = predict(lassos.fullinteraction, X.fullinteraction)
lassosfullinteraction.trainRMSE = RMSE(train.df$W.L..next_year, yhats.fullinteraction.train) # train RM
lassosfullinteraction.trainR2 = R2(train.df$W.L..next_year, yhats.fullinteraction.train) # train R2

yhats.fullinteraction.test = predict(lassos.fullinteraction, X.fullinteraction.test)
#plot(RMSE.lassos.fullinteraction.test~log(lassos.fullinteraction$lambda, 10), type='l')
lassosfullinteraction.testRMSE = RMSE(test.df$W.L..next_year, yhats.fullinteraction.test) # train RMSE
lassosfullinteraction.testR2 = R2(test.df$W.L..next_year, yhats.fullinteraction.test) # train R2
```

```
# Stepwise
lm.step = step(lm.full, scope=c(lower=formula(W.L..next_year~1),
                                upper=lm.fullinteraction), trace=0, direction="both")
formula(lm.step)
```

```
## W.L..next_year ~ Age.bat + AB + H.bat + X3B + BB.bat + OBP +
##     OPS + GDP + SH + Age.pitch + GF + IP + IBB.pitch + SO.pitch +
##     BF + HR9 + Inn + Ch + PO + A + tot_fa_war3 + num_fas
```

```
lmstep.trainRMSE = RMSE(train.df$W.L..next_year, predict(lm.step, newdata=train.df))
lmstep.testRMSE = RMSE(test.df$W.L..next_year, predict(lm.step, newdata=test.df))
lmstep.trainR2 = R2(train.df$W.L..next_year, predict(lm.step, newdata=train.df))
lmstep.testR2 = R2(test.df$W.L..next_year, predict(lm.step, newdata=test.df))
```

```
# model comparison
RMSE.df = data.frame(trainRMSE = c(baseline.trainRMSE,
                                   lmfull.trainRMSE,
                                   lmfullinteraction.trainRMSE,
                                   ridgesfull.trainRMSE,
                                   ridgesfullinteraction.trainRMSE,
                                   lassosfull.trainRMSE,
                                   lassosfullinteraction.trainRMSE,
                                   lmstep.trainRMSE),
                     testRMSE = c(baseline.testRMSE,
                                   lmfull.testRMSE,
                                   lmfullinteraction.testRMSE,
```

```
                                ridgesfull.testRMSE,
                                ridgesfullinteraction.testRMSE,
                                lassosfull.testRMSE,
                                lassosfullinteraction.testRMSE,
                                lmstep.testRMSE),
                    trainR2 = c(baseline.trainR2,
                                lmfull.trainR2,
                                lmfullinteraction.trainR2,
                                ridgesfull.trainR2,
                                ridgesfullinteraction.trainR2,
                                lassosfull.trainR2,
                                lassosfullinteraction.trainR2,
                                lmstep.trainR2),
                    testR2 = c(baseline.testR2,
                                lmfull.testR2,
                                lmfullinteraction.testR2,
                                ridgesfull.testR2,
                                ridgesfullinteraction.testR2,
                                lassosfull.testR2,
                                lassosfullinteraction.testR2,
                                lmstep.testR2))
rownames(RMSE.df) <- c("baseline", "full", "full interaction",
                       "ridge full", "ridge full interaction",
                       "lasso full", "lasso full interaction",
                       "step")
RMSE.df
```

```
##                            trainRMSE    testRMSE   trainR2        testR2
## baseline                6.519237e+00    6.071293 0.2782818  2.510428e-01
## full                    5.876126e+00    6.359129 0.4136509  1.783443e-01
## full interaction        1.285132e-08 1094.710680 1.0000000 -2.434868e+04
## ridge full              6.063804e+00    5.954488 0.3755980  2.795838e-01
## ridge full interaction  6.046672e+00    6.071837 0.3791212  2.509086e-01
## lasso full              6.045262e+00    5.975223 0.3794106  2.745578e-01
## lasso full interaction  5.960361e+00    6.008083 0.3967197  2.665567e-01
## step                    5.955305e+00    6.291723 0.3977427  1.956708e-01
```

Decision Tree/Random Forest

```
set.seed(139)
library(rpart)

RMSE = function(y,yhat){
  return(sqrt(mean((y-yhat)^2)))
}

test.df = subset(test.df, test.df$Tm != 'CLE')
tree1 = rpart(formula(lm.full),data=train.df, control = list(minsplit=1,cp=0,maxdepth=20))
yhat.tree1.train = predict(tree1)
yhat.tree1.test = predict(tree1, newdata = test.df)
RMSE.tree1.train = RMSE(train.df$W.L..next_year,yhat.tree1.train)
```

```
RMSE.tree1.test = RMSE(test.df$W.L..next_year,yhat.tree1.test)
data.frame(train=RMSE.tree1.train,test=RMSE.tree1.test)
```

```
##      train     test
## 1 4.049359 7.899947
```

```
best.cp = tree1$cptable[,"CP"][which.min(tree1$cptable[,"xerror"])]
tree2 = prune(tree1,best.cp)
yhat.tree2.train = predict(tree2)
yhat.tree2.test = predict(tree2,newdata=test.df)
RMSE.tree2.train = RMSE(train.df$W.L..next_year,yhat.tree2.train)
RMSE.tree2.test = RMSE(test.df$W.L..next_year,yhat.tree2.test)
data.frame(train=RMSE.tree2.train,test=RMSE.tree2.test)
```

```
##      train     test
## 1 7.200247 6.589295
```

```
tree3 = rpart(W.L..next_year~W.L..same_year + Age.pitch + WHIP,
              data=train.df, control = list(minsplit=1, cp=0, maxdepth=20))
yhat.tree3.train = predict(tree3)
yhat.tree3.test = predict(tree3, newdata = test.df)
RMSE.tree3.train = RMSE(train.df$W.L..next_year,yhat.tree3.train)
RMSE.tree3.test = RMSE(test.df$W.L..next_year,yhat.tree3.test)
data.frame(train=RMSE.tree3.train,test=RMSE.tree3.test)
```

```
##     train    test
## 1 5.40947 6.97301
```

```
best.cp = tree3$cptable[,"CP"][which.min(tree3$cptable[,"xerror"])]
tree4 = prune(tree3,best.cp)
yhat.tree4.train = predict(tree4)
yhat.tree4.test = predict(tree4,newdata=test.df)
RMSE.tree4.train = RMSE(train.df$W.L..next_year,yhat.tree4.train)
RMSE.tree4.test = RMSE(test.df$W.L..next_year,yhat.tree4.test)
data.frame(train=RMSE.tree4.train,test=RMSE.tree4.test)
```

```
##      train     test
## 1 6.694751 6.631472
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.2.2
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
set.seed(139)
maxnodes = c(100,200,500)
ntree= 200
rmses.bag = rep(NA,length(maxnodes))
bestRMSE = sd(train.df$W.L..next_year)

for(i in 1:length(maxnodes)){
  bagtemp = randomForest(formula(lm.full),data=train.df,
                         mtry=56, maxnodes=maxnodes[i], ntree=ntree)
  rmses.bag[i]=RMSE(train.df$W.L..next_year, bagtemp$predicted)
  if(rmses.bag[i]<bestRMSE){
    best_maxnodes = maxnodes[i]
    bestRMSE=rmses.bag[i]
    bag=bagtemp
  }
}
data.frame(maxnodes=maxnodes, RMSE=rmses.bag)
```

```
##   maxnodes     RMSE
## 1      100 6.659953
## 2      200 6.659720
## 3      500 6.656250
```

```
yhat.bag.train = predict(bag)
yhat.bag.test = predict(bag, newdata = test.df)
RMSE.bag.train = RMSE(train.df$W.L..next_year,yhat.bag.train)
RMSE.bag.test = RMSE(test.df$W.L..next_year,yhat.bag.test)
data.frame(train=RMSE.bag.train,test=RMSE.bag.test)
```

```
##     train    test
## 1 6.65625 6.16641
```

```
library(randomForest)
library(varImp)
```

```
## Warning: package 'varImp' was built under R version 4.2.2

## Loading required package: measures

## Warning: package 'measures' was built under R version 4.2.2

##
## Attaching package: 'measures'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     RMSE

## The following objects are masked from 'package:caret':
##
##     MAE, RMSE

## Loading required package: party

## Warning: package 'party' was built under R version 4.2.2

## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Warning: package 'strucchange' was built under R version 4.2.2

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.2.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

## Warning: package 'sandwich' was built under R version 4.2.2

##
## Attaching package: 'varImp'

## The following object is masked from 'package:caret':
##
##     varImp
```

```
set.seed(139)
maxnodes = c(100,200,500)
mtry = c(15, 25, 35, 45, 55)
ntree=200
pars = expand.grid(maxnodes=maxnodes,mtry=mtry)
RMSEs = rep(NA,nrow(pars))
bestRMSE = sd(train.df$W.L..next_year)

for(i in 1:nrow(pars)){
  rftemp = randomForest(formula(lm.full),data=train.df,
                        mtry=pars$mtry[i], maxnodes=pars$maxnodes[i], ntree=ntree)
  RMSEs[i]=RMSE(train.df$W.L..next_year, rftemp$predicted)
  if(RMSEs[i]<bestRMSE){
    best_maxnodes = maxnodes[i]
    bestRMSE=RMSEs[i]
    rf1=rftemp
  }
}
data.frame(maxnodes=pars$maxnodes,mtry=pars$mtry,RMSE=RMSEs)
```

```
##    maxnodes mtry     RMSE
## 1       100   15 6.639861
## 2       200   15 6.604967
## 3       500   15 6.668886
## 4       100   25 6.660616
## 5       200   25 6.659093
## 6       500   25 6.609204
## 7       100   35 6.643866
## 8       200   35 6.596726
## 9       500   35 6.649860
## 10      100   45 6.607546
## 11      200   45 6.613605
## 12      500   45 6.662391
## 13      100   55 6.653295
## 14      200   55 6.713727
## 15      500   55 6.667542
```

```
pars[which(RMSEs==bestRMSE),]
```

```
##   maxnodes mtry
## 8      200   35
```

```
yhat.rf1.train = predict(rf1)
yhat.rf1.test = predict(rf1, newdata = test.df)
RMSE.rf1.train = RMSE(train.df$W.L..next_year,yhat.rf1.train)
RMSE.rf1.test = RMSE(test.df$W.L..next_year,yhat.rf1.test)
data.frame(train=RMSE.tree1.train,test=RMSE.rf1.test)
```

```
##      train     test
## 1 4.049359 6.175025
```

```
importance(rf1)
```

```
##                 IncNodePurity
## Age.bat              492.1182
## PA                   139.7609
## AB                   202.9089
## H.bat                140.0418
## X2B                  263.8188
## X3B                  400.9269
## HR.bat               535.2130
## SB                   336.1993
## CS                   327.7132
## BB.bat               951.2127
## SO.bat               380.9627
## BA                   291.6270
## OBP                  650.7102
## SLG                  415.5432
## OPS                  764.9419
## OPSplus              655.5048
## TB                   146.2555
## GDP                  359.3066
## HBP.bat              520.0794
## SH                   393.7992
## SF                   313.5331
## IBB.bat              894.7904
## Age.pitch            973.4202
## W.L..same_year      2545.3689
## GF                   262.7406
## SHO                  499.3218
## SV                   457.4116
## IP                   263.7419
## H.pitch              341.5894
## HR.pitch             403.8534
## BB.pitch             370.0386
## IBB.pitch            366.1679
## SO.pitch             818.3978
## HBP.pitch            326.1395
## BK                   530.2902
## WP                   416.5869
## BF                   313.5373
## FIP                 1106.0739
## WHIP                2004.6213
## H9                  1297.5072
## HR9                  602.5493
## BB9                  522.2450
## SO9                  378.7967
## SO.W                 413.0191
## G                    406.0180
## Inn                  324.5931
## Ch                   530.7242
## PO                   415.5982
## A                    357.5136
## E                    532.1435
```

```
## DP                   419.2013
## Fld.                 451.2475
## RF.9                 411.1745
## RF.G                 397.5741
## tot_fa_war3         1403.6797
## num_fas              351.7310
```

```
library(randomForest)
set.seed(139)
maxnodes = c(100,200,500)
mtry = c(1,2,3)
ntree=200
pars = expand.grid(maxnodes=maxnodes,mtry=mtry)
RMSEs = rep(NA,nrow(pars))
bestRMSE = sd(train.df$W.L..next_year)

for(i in 1:nrow(pars)){
  rftemp = randomForest(W.L..next_year ~ W.L..same_year + Age.pitch + WHIP, data=train.df,
                        mtry=pars$mtry[i], maxnodes=pars$maxnodes[i], ntree=ntree)
  RMSEs[i]=RMSE(train.df$W.L..next_year, rftemp$predicted)
  if(RMSEs[i]<bestRMSE){
    best_maxnodes = maxnodes[i]
    bestRMSE=RMSEs[i]
    rf2=rftemp
  }
}
data.frame(maxnodes=pars$maxnodes,mtry=pars$mtry,RMSE=RMSEs)
```

```
##   maxnodes mtry     RMSE
## 1      100    1 6.916935
## 2      200    1 7.004427
## 3      500    1 6.948551
## 4      100    2 6.987361
## 5      200    2 7.059185
## 6      500    2 7.066759
## 7      100    3 7.031486
## 8      200    3 7.123478
## 9      500    3 7.099217
```

```
pars[which(RMSEs==bestRMSE),]
```

```
##   maxnodes mtry
## 1      100    1
```

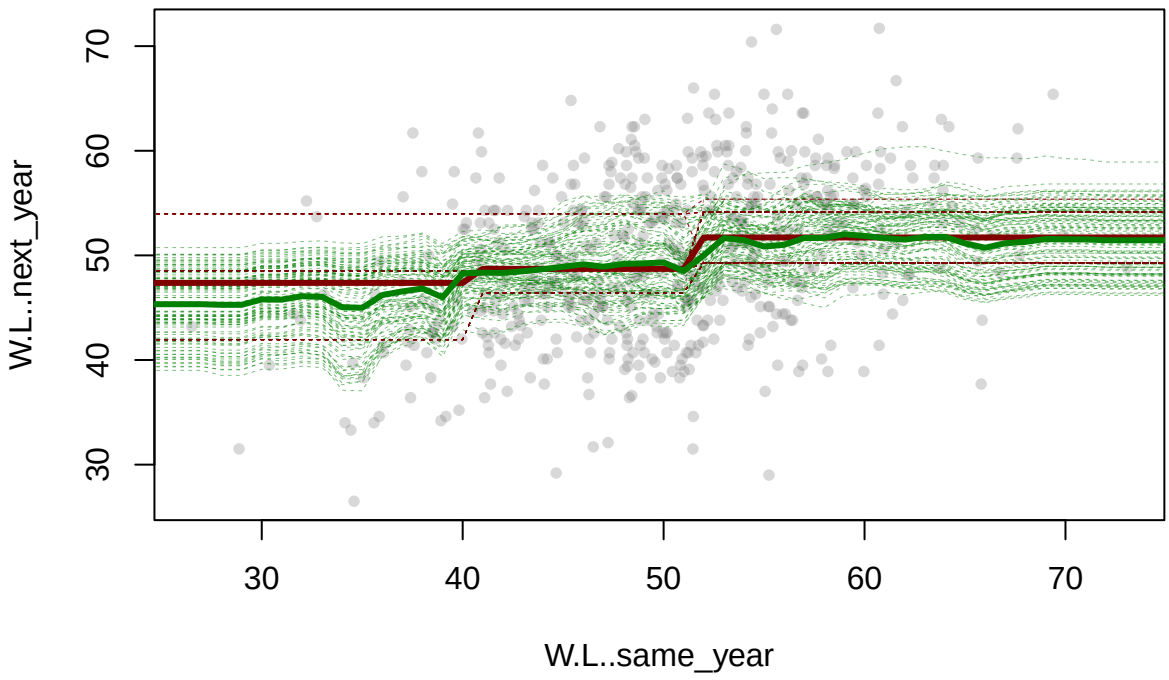
```
yhat.rf2.train = predict(rf2)
yhat.rf2.test = predict(rf2, newdata = test.df)
RMSE.rf2.train = RMSE(train.df$W.L..next_year,yhat.rf2.train)
RMSE.rf2.test = RMSE(test.df$W.L..next_year,yhat.rf2.test)
data.frame(train=RMSE.tree1.train,test=RMSE.rf2.test)
```

```
##      train    test
## 1 4.049359 6.36324
```

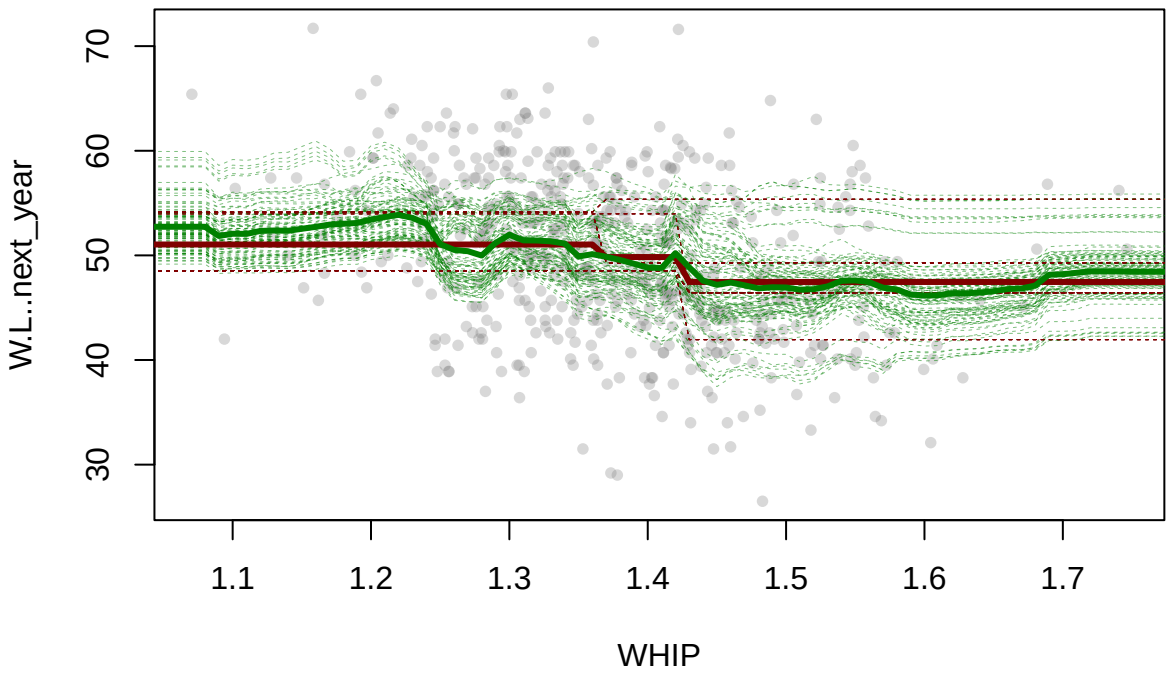
```
importance(rf2)
```

```
##                 IncNodePurity
## W.L..same_year      8095.005
## Age.pitch           6832.835
## WHIP                7926.658
```

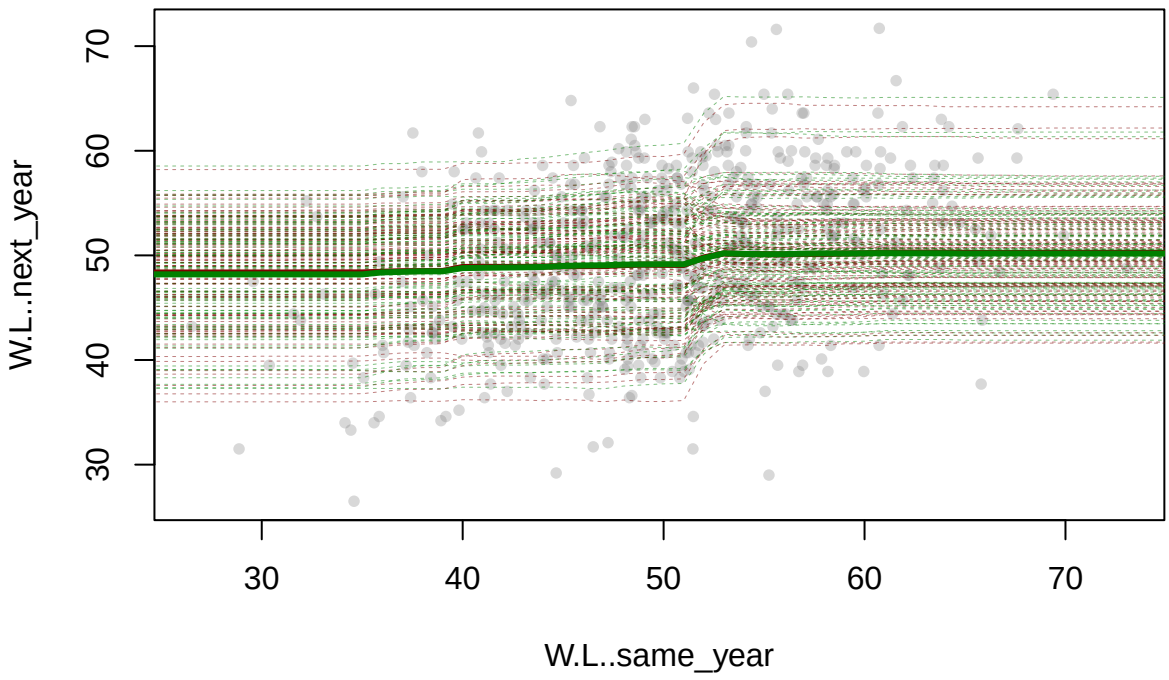
```
set.seed(139)
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(0,100,1)
plot(W.L..next_year~W.L..same_year, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
  rows$W.L..same_year=dummyx
  yhat = predict(tree4,new=rows)
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
  yhat.rf = predict(rf2,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```



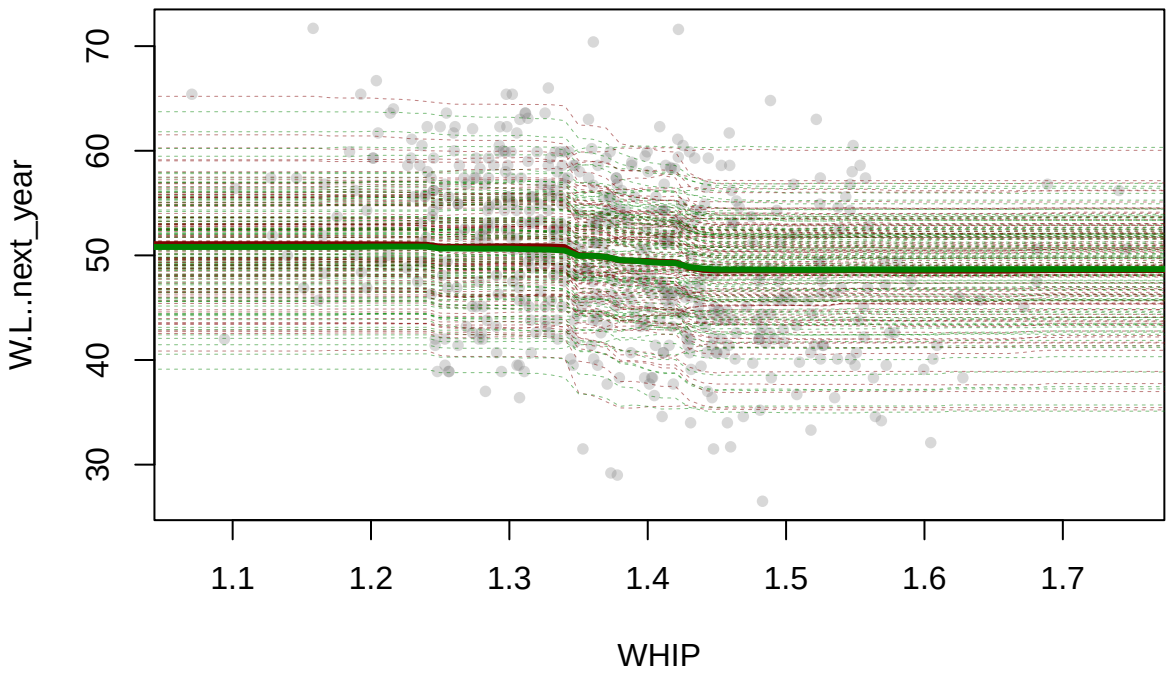

```
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(1,2,.01)
plot(W.L..next_year~WHIP, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
  rows$WHIP=dummyx
  yhat = predict(tree4,new=rows)
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
  yhat.rf = predict(rf2,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```



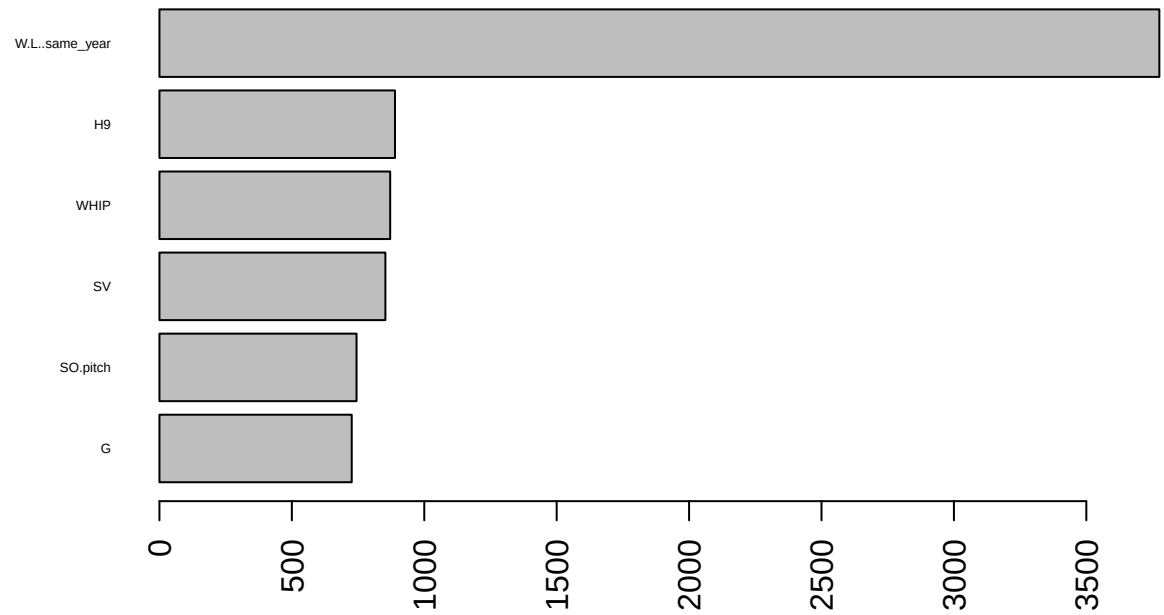

```
set.seed(139)
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(0,100,1)
plot(W.L..next_year~W.L..same_year, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
  rows$W.L..same_year=dummyx
  yhat = predict(bag,new=rows)
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
  yhat.rf = predict(rf1,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```



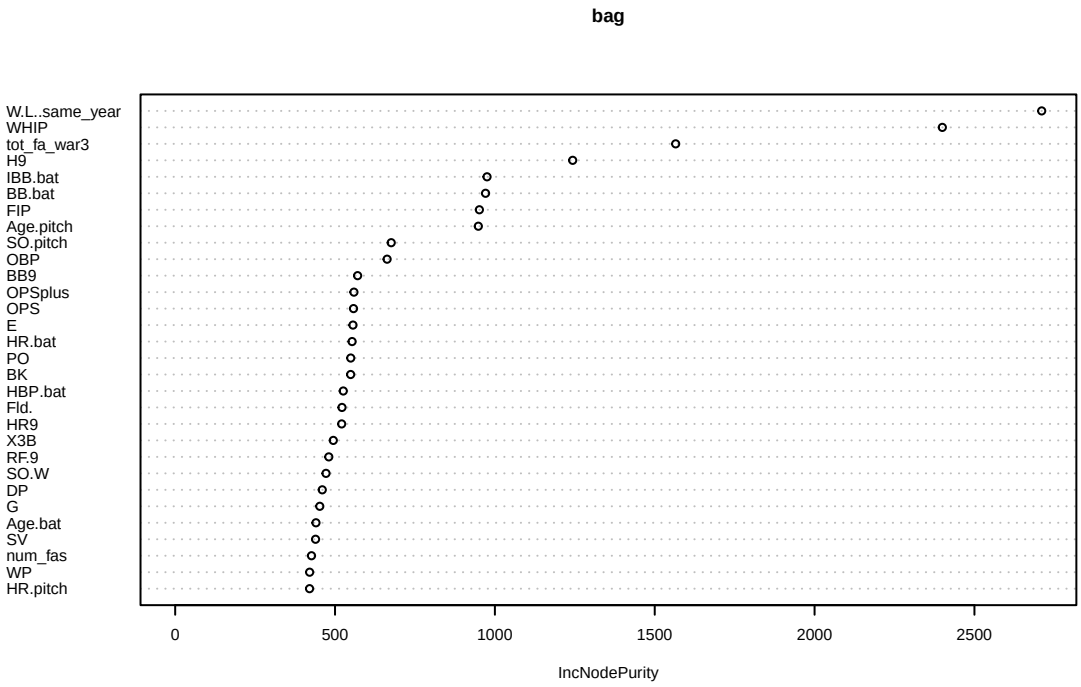

```
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(1,2,.01)
plot(W.L..next_year~WHIP, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
  rows$WHIP=dummyx
  yhat = predict(bag,new=rows)
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
  yhat.rf = predict(rf1,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```



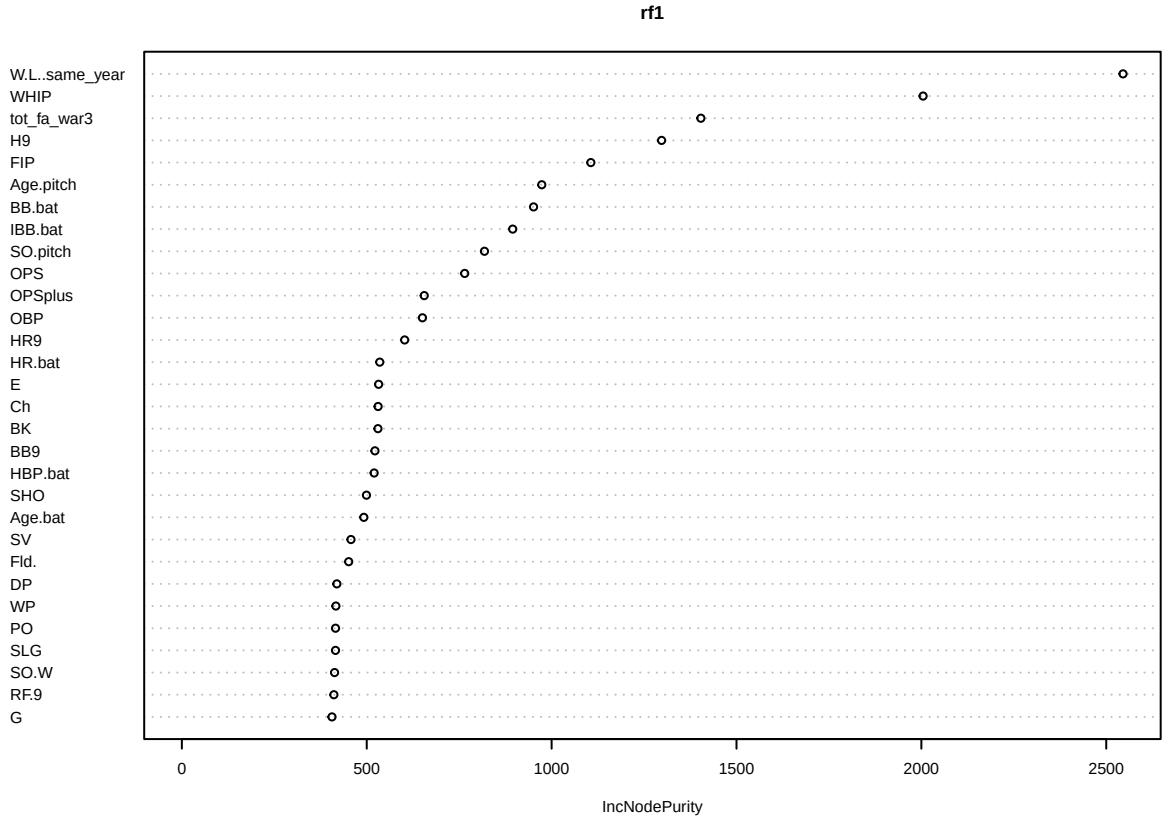

```
barplot(sort(tree2$variable.importance),horiz = T,las=2,cex.names = 0.4, main='Variable Importance for
```



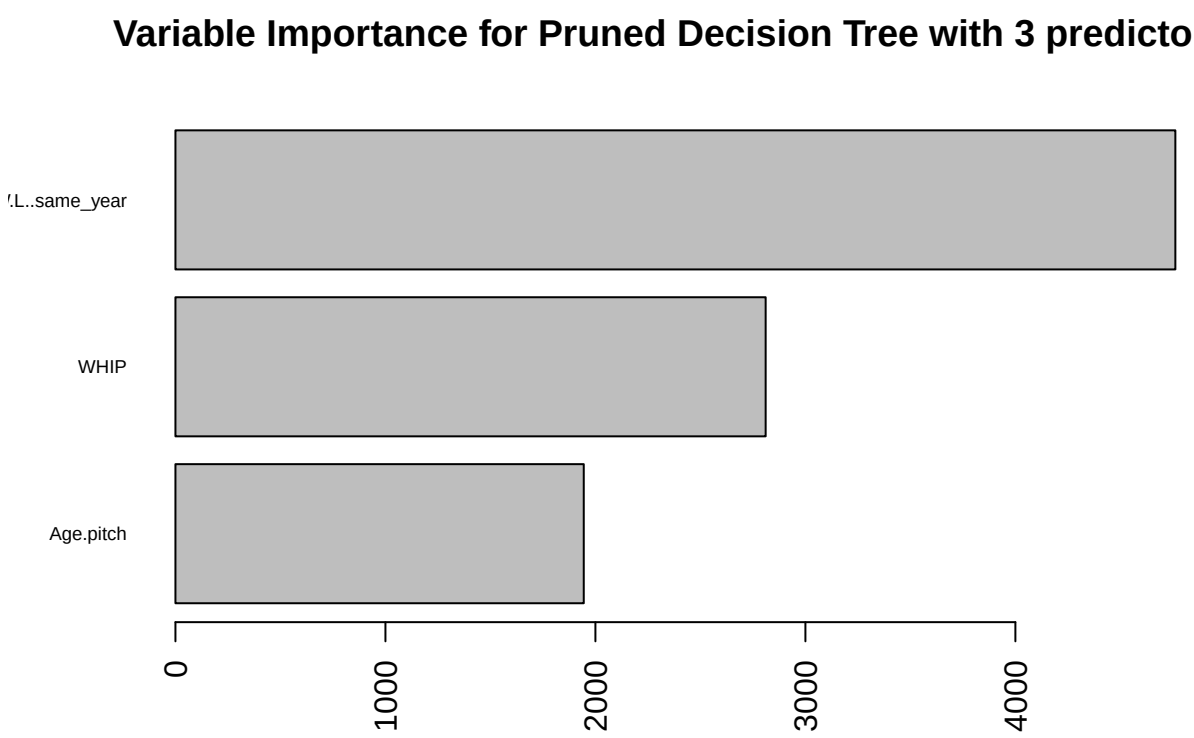

```
varImpPlot(bag, cex=0.5)
```



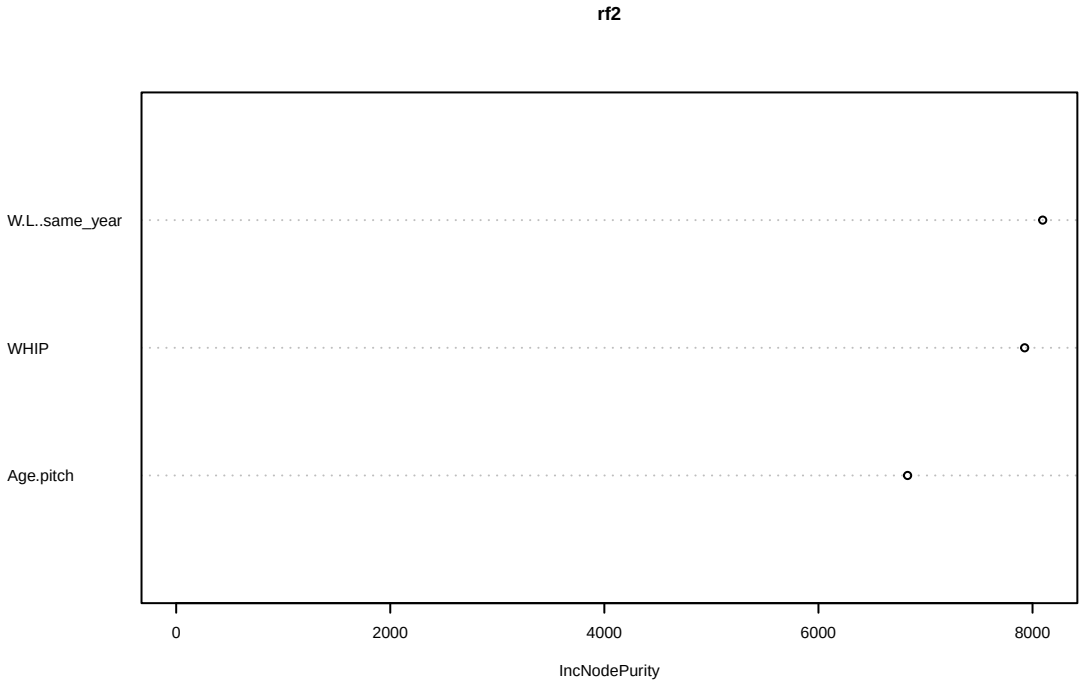

```
varImpPlot(rf1,cex=0.5)
```




```
barplot(sort(tree4$variable.importance),horiz = T,las=2,cex.names = 0.6, main='Variable Importance for
```




```
varImpPlot(rf2, cex=0.5)
```




```
library(lme4)
```

```
## Warning: package 'lme4' was built under R version 4.2.2
```

```
##
## Attaching package: 'lme4'
```

```
## The following object is masked from 'package:modeltools':
##
##     refit
```

```
set.seed(139)

# for (i in 1997:2022){
#   lmer_model <- lmer(team_data[[i]]$W.L.~poly(team_data[[i]]$BatAge, 2, raw = TRUE) + (1 + poly(team_
#   summary(lmer_model)
# }

lmer_model <- lmer(train.df$W.L..next_year ~ poly(train.df$Age.bat, 2, raw = FALSE) + (1 + poly(train.d
summary(lmer_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: train.df$W.L..next_year ~ poly(train.df$Age.bat, 2, raw = FALSE) +
##     ((1 | train.df$Tm) + (0 + poly(train.df$Age.bat, 2, raw = FALSE) |
```

```
##         train.df$Tm))
##
## REML criterion at convergence: 3613.5
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.89140 -0.66082  0.00841  0.67136  3.05598
##
## Random effects:
##  Groups        Name                                     Variance Std.Dev. Corr
##  train.df.Tm   (Intercept)                               12.93    3.595
##  train.df.Tm.1 poly(train.df$Age.bat, 2, raw = FALSE)1 853.35   29.212
##                poly(train.df$Age.bat, 2, raw = FALSE)2 366.15   19.135   -1.00
##  Residual                                               45.04    6.711
## Number of obs: 536, groups:  train.df$Tm, 29
##
## Fixed effects:
##                                         Estimate Std. Error t value
## (Intercept)                              49.7990     0.7377  67.503
## poly(train.df$Age.bat, 2, raw = FALSE)1  -3.2378     9.6347  -0.336
## poly(train.df$Age.bat, 2, raw = FALSE)2  10.0864     8.5488   1.180
##
## Correlation of Fixed Effects:
##                     (Intr) p(.$A.,2,r=FALSE)1
## p(.$A.,2,r=FALSE)1 -0.017
## p(.$A.,2,r=FALSE)2  0.038 -0.296
```

```
lmer_model <- lmer(train.df$W.L..next_year ~ poly(train.df$BA, 2, raw = FALSE) + (1 + poly(train.df$BA,
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00229477 (tol = 0.002, component 1)
```

```
summary(lmer_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: train.df$W.L..next_year ~ poly(train.df$BA, 2, raw = FALSE) +
##     ((1 | train.df$Tm) + (0 + poly(train.df$BA, 2, raw = FALSE) |
##         train.df$Tm))
##
## REML criterion at convergence: 3603.7
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.7908 -0.6490  0.0066  0.6976  2.9475
##
## Random effects:
##  Groups        Name                                 Variance Std.Dev. Corr
##  train.df.Tm   (Intercept)                           13.169   3.629
##  train.df.Tm.1 poly(train.df$BA, 2, raw = FALSE)1 274.432  16.566
##                poly(train.df$BA, 2, raw = FALSE)2   5.864   2.422   1.00
##  Residual                                           45.196   6.723
## Number of obs: 536, groups:  train.df$Tm, 29
##
```

```
## Fixed effects:
##                                  Estimate Std. Error t value
## (Intercept)                       49.7826     0.7446  66.862
## poly(train.df$BA, 2, raw = FALSE)1  34.7817     9.5007   3.661
## poly(train.df$BA, 2, raw = FALSE)2   3.3898     7.2775   0.466
##
## Correlation of Fixed Effects:
##                    (Intr) p(.$BA,2,r=FALSE)1
## p(.$BA,2,r=FALSE)1 0.018
## p(.$BA,2,r=FALSE)2 0.036  0.036
## optimizer (nloptwrap) convergence code: 0 (OK)
## Model failed to converge with max|grad| = 0.00229477 (tol = 0.002, component 1)
```

```
# lmer_model <- lmer(W.L..next_year ~ Age.bat + PA + AB + H.bat + X2B + X3B +
                # HR.bat + SB + CS + BB.bat + SO.bat + BA + OBP + SLG + OPS + OPSplus +
                # TB + GDP + HBP.bat + SH + SF + IBB.bat + Age.pitch + W.L..same_year +
                # GF + SHO + SV + IP + H.pitch + HR.pitch +
                # BB.pitch + IBB.pitch + SO.pitch + HBP.pitch + BK + WP + BF +
                # FIP + WHIP + H9 + HR9 + BB9 + SO9 + SO.W +
                # G + Inn + Ch + PO + A + E + DP + Fld. +
                # RF.9 + RF.G + tot_fa_war3 + num_fas || Tm, data = train.df, verbose=TRUE)
```

```
summary(lmer_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: train.df$W.L..next_year ~ poly(train.df$BA, 2, raw = FALSE) +
##     ((1 | train.df$Tm) + (0 + poly(train.df$BA, 2, raw = FALSE) |
##         train.df$Tm))
##
## REML criterion at convergence: 3603.7
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.7908 -0.6490  0.0066  0.6976  2.9475
##
## Random effects:
##  Groups        Name                                Variance Std.Dev. Corr
##  train.df.Tm   (Intercept)                          13.169   3.629
##  train.df.Tm.1 poly(train.df$BA, 2, raw = FALSE)1 274.432  16.566
##                poly(train.df$BA, 2, raw = FALSE)2   5.864   2.422   1.00
##  Residual                                          45.196   6.723
## Number of obs: 536, groups:  train.df$Tm, 29
##
## Fixed effects:
##                                  Estimate Std. Error t value
## (Intercept)                       49.7826     0.7446  66.862
## poly(train.df$BA, 2, raw = FALSE)1  34.7817     9.5007   3.661
## poly(train.df$BA, 2, raw = FALSE)2   3.3898     7.2775   0.466
##
## Correlation of Fixed Effects:
##                    (Intr) p(.$BA,2,r=FALSE)1
## p(.$BA,2,r=FALSE)1 0.018
## p(.$BA,2,r=FALSE)2 0.036  0.036
```

```
## optimizer (nloptwrap) convergence code: 0 (OK)
## Model failed to converge with max|grad| = 0.00229477 (tol = 0.002, component 1)
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + (1 + WHIP + W.L..same_year +
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 1.18788 (tol = 0.002, component 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unider
##  - Rescale variables?;Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?
```

```
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.122982
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 5.963312
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch | Tm, data = train.df)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues
```

```
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.095754
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.139856
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + tot_fa_war3 | Tm, data = tra
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues
```

```
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.911676
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.535433
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + H9 | Tm, data = train.df)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.840416
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.235415
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + H9 + (1 + WHIP + W.L..same_y
```

```
## boundary (singular) fit: see help('isSingular')
```

```
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.879447
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.039918
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ Age.bat + PA + AB | Tm, data = train.df)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 2 negative eigenvalues
```

```
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.529421
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.270939
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ Age.bat + PA + AB + (1 + Age.bat + PA + AB | Tm) , data = train.
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 2 negative eigenvalues
```

```
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.157979
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.353319
```